Model Transformations with Tom

Jean-Christophe Bach[1], Xavier Crégut[2],
Pierre-Étienne Moreau[1] and Marc Pantel[2]

[1]INRIA, LORIA & Université de Lorraine, France
[2]INPT-IRIT & Université de Toulouse, France

LDTA 2012

Context and objectives

- Wide use of MDE: importance of transformations to automate repetitive development tasks
- Increasing trust in software: writing qualified models transformations (quarteFt project)
- Taking advantage of general purpose languages features and having a dedicated language to easily transform models

Our solution

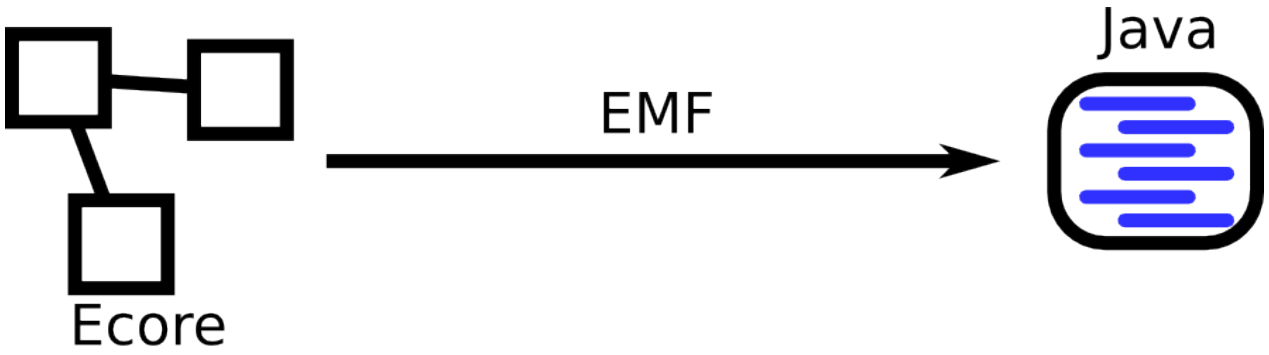
- Two main approaches to write models transformations
  - using a general purpose language: full Java + EMF
  - using a Domain Specific Language (DSL): ATL, Kermeta, *etc.*
- Our solution: being between the two approches by using tools
  - Tom language: extending Java
  - Tom-EMF: representing EMF models with Tom

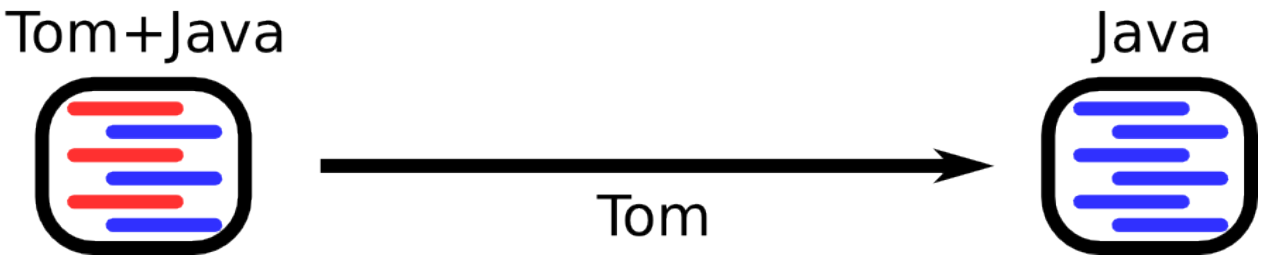
$\Rightarrow$ a first step to a high-level models transformations language integrated into Java

Tom: an extension of general purpose languages

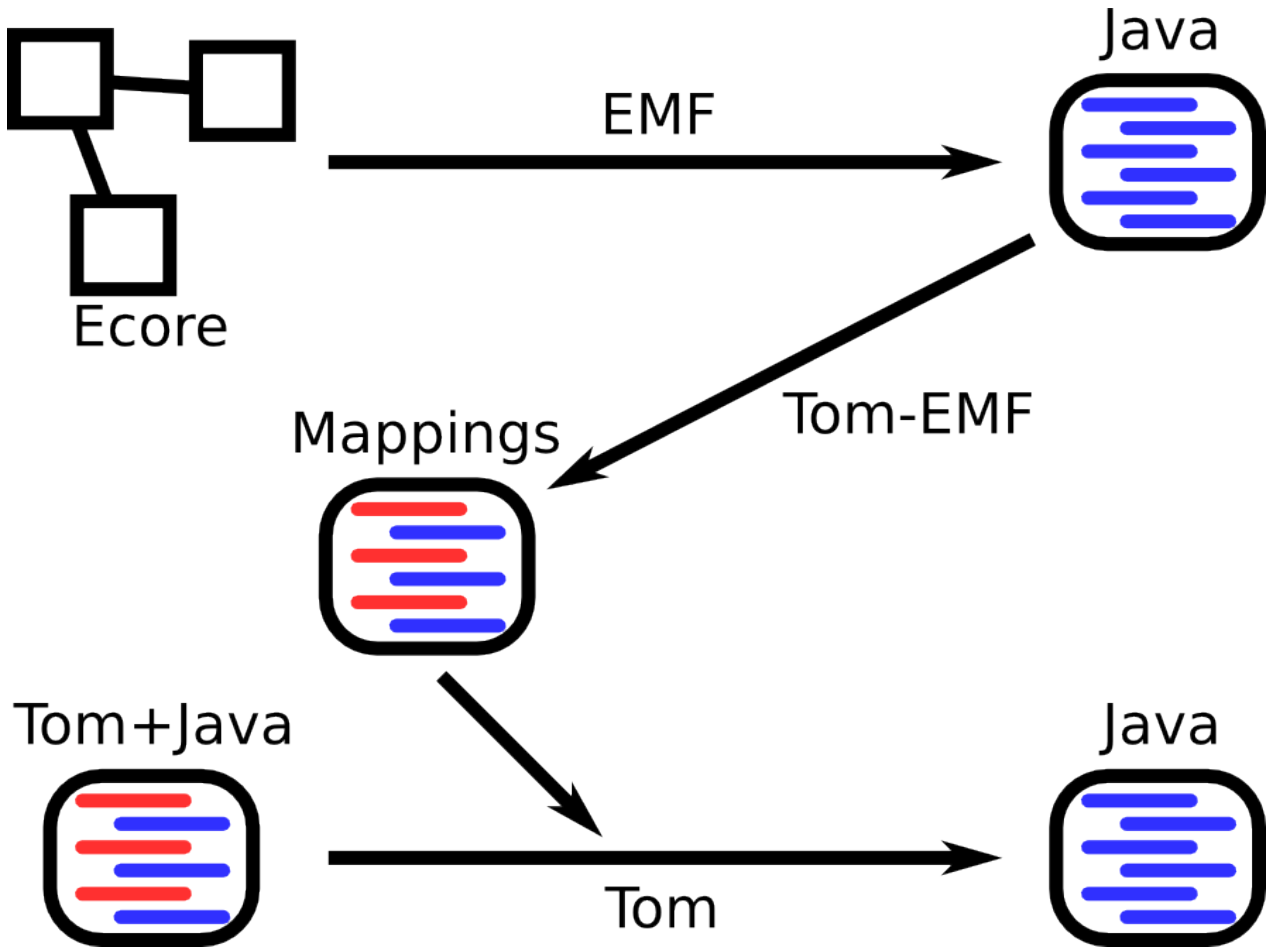
- Adding features to Java
  - pattern-matching: generalization of "switch-case" construct
  - mappings: representation of Java objects as Tom terms
  - strategies: increase of control over rewriting rules application
- A good way to manipulate tree structures

Tom-EMF: handling EMF with Tom

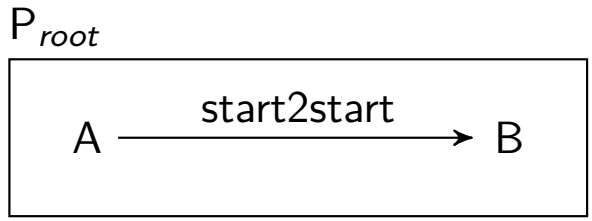




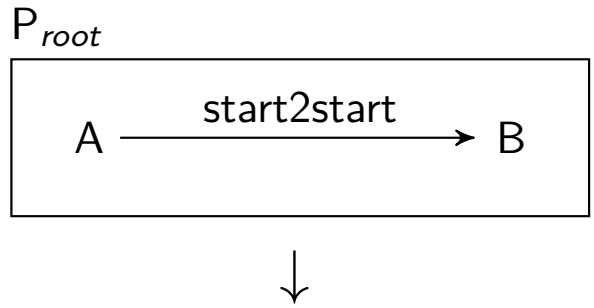
Tom-EMF: handling EMF with Tom



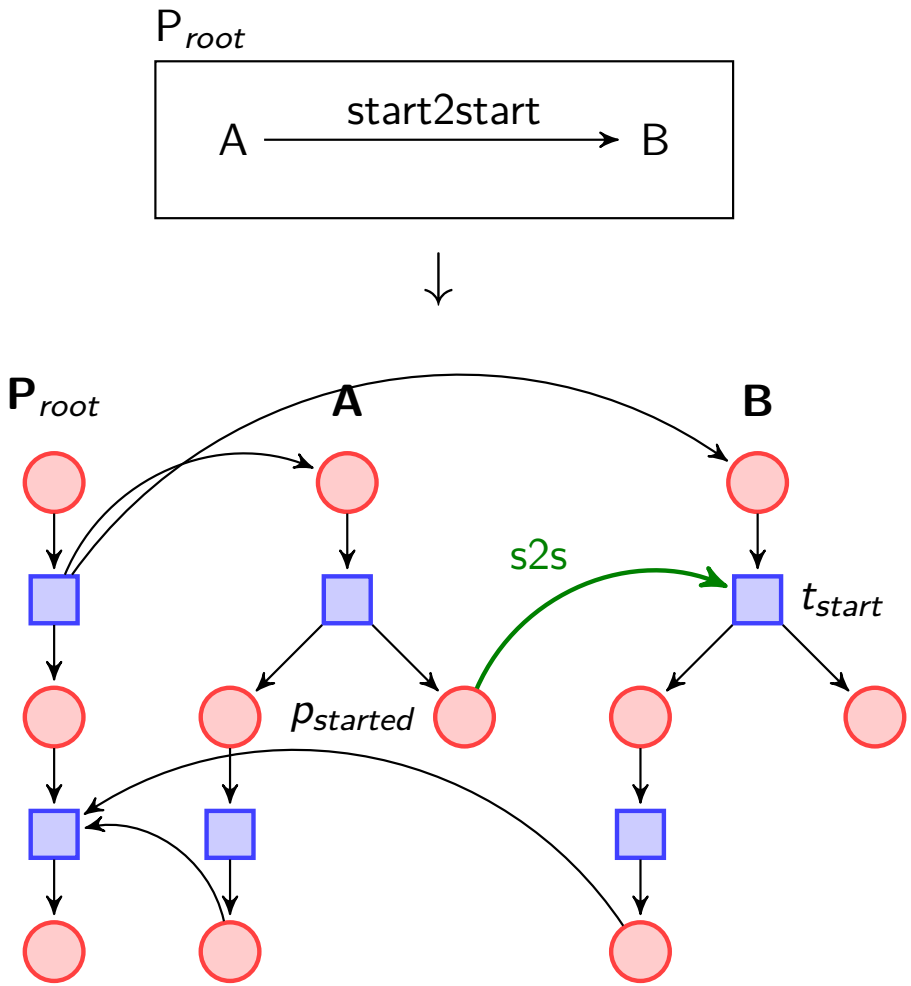
Example of transformation: SimplePDLToPetriNet



Example of transformation: SimplePDLToPetriNet



Example of transformation: SimplePDLToPetriNet

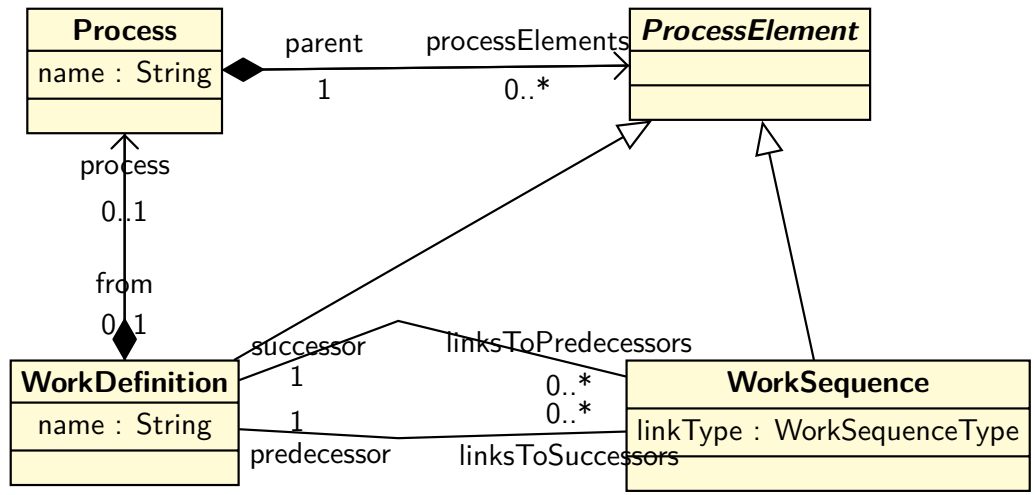


Method to implement this transformation

- Before beginning
  - defining source and target metamodels with EMF
  - generating corresponding Java code
  - generating Tom-EMF mappings
- Transformation itself
  - decomposing the transformation into elementary ones
  - writing a strategy for each atomic transformation
  - applying all of these simple transformations

Application to the SimplePDLToPetriNet use case

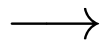
- Three main structures in a SimplePDL process
  - `Process`
  - `WorkDefinition`
  - `WorkSequence`

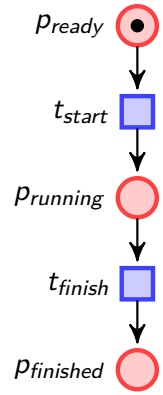


- Let's write a simple transformation (strategy) for each one:
  - `Process2PetriNet`
  - `WorkDefinition2PetriNet`
  - `WorkSequence2PetriNet`

Elementary transformations (1/2)

- Process

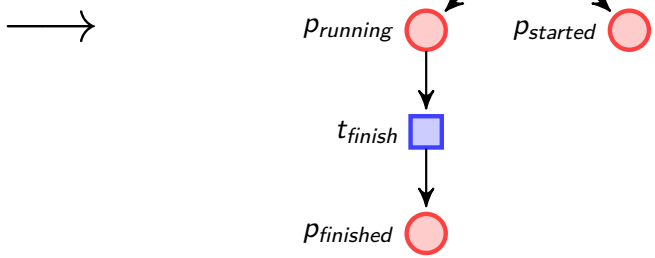





```
%strategy Process2PetriNet(pn:PetriNet) extends Identity () {
  visit Process {
    p@Process[name=name,from=from] -> {
      Node p_ready = `Place(name + "_ready", pn);
      Node t_start = `Transition(name + "_start");
       ... // creation of p_running, t_finish, and p_finished
      `Arc(t_start, p_ready, pn,normal());
       ... // creation of the 3 remainings arcs
    }
  }
}
```

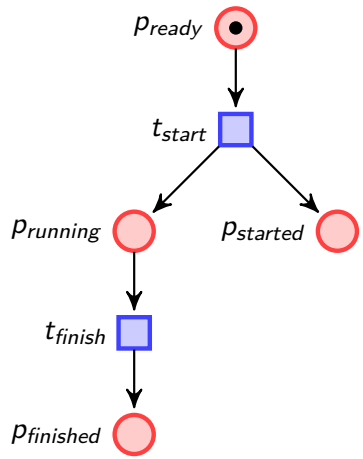
Elementary transformations (2/2)

- WorkDefinition

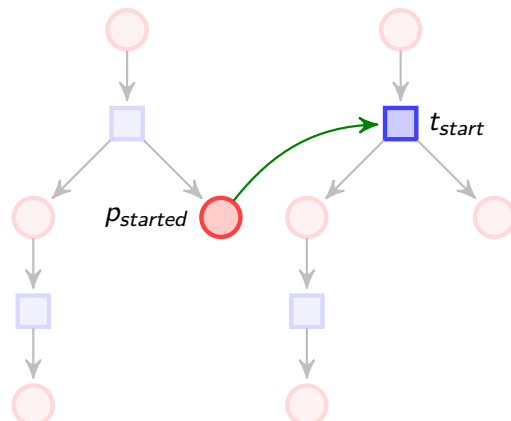


Elementary transformations (2/2)

- WorkDefinition



- WorkSequence



How to express what does not yet exist?

`WorkSequence[pred=A,succ=B]` $\longrightarrow$

```
create  Arc
  from  Image(A).p_started
  to    Image(B).t_start
;
```

How to express what does not yet exist?

`WorkSequence[pred=A,succc=B]` $\longrightarrow$

```
create  Arc
  from  Image(A).p_started
  to    Image(B).t_start
;
```

$\Rightarrow$ by enriching the target metamodel: addition of "resolve elements"

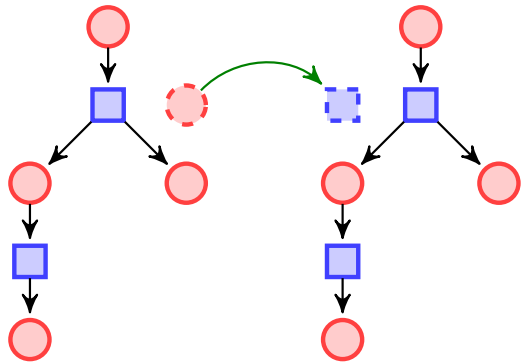
How to express what does not yet exist?

`WorkSequence[pred=A,succ=B]` $\longrightarrow$

```
create  Arc
  from  Image(A).p_started
  to    Image(B).t_start
;
```

$\Rightarrow$ by enriching the target metamodel: addition of "resolve elements"
and
by adding a new strategy: "Resolve"

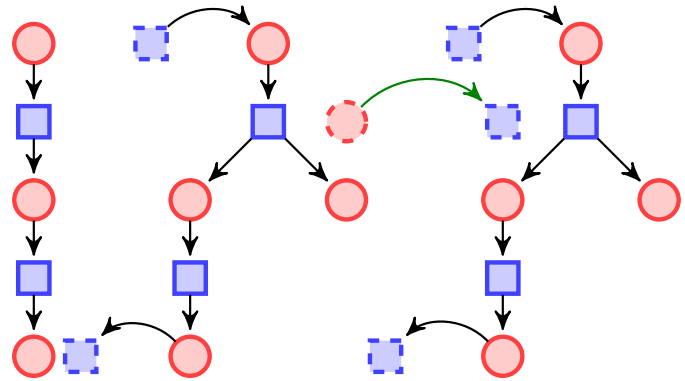
Applying strategies



- Tom strategy representing the whole transformation:

```
‘Sequence(TopDown(WorkSequence2PetriNet(pn)),
          TopDown(WorkDefinition2PetriNet(pn)));
```

Applying strategies



- Tom strategy representing the whole transformation:

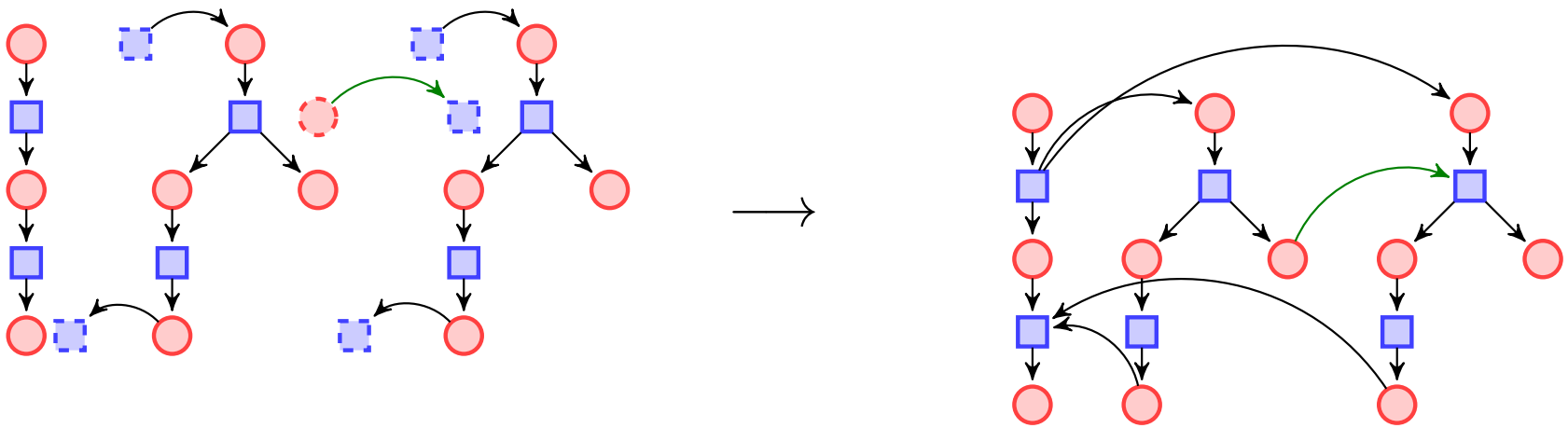
```
‘Sequence(TopDown(WorkSequence2PetriNet(pn)),
          TopDown(Process2PetriNet(pn)),
          TopDown(WorkDefinition2PetriNet(pn)));
```

Reconnecting intermediate results

- A new strategy applied at the end of the transformation: `‘TopDown(Resolve(pn));`

Reconnecting intermediate results

- A new strategy applied at the end of the transformation:
  `'TopDown(Resolve(pn));`
- What does the `Resolve` strategy do?
  - it finds all resolve elements
  - it replaces them by the corresponding target elements
    $\Rightarrow$ reconnects partial results to build the final one



Conclusion

- Tools to represent, manipulate and transform EMF models in general purpose languages
- A methodology to use Tom for models transformations
- A first step to a high-level embedded transformation language:
  - **it will be the basis of new Tom constructs**
- More about Tom: `http://tom.loria.fr`
- Demo available until the end of LDTA: ask me or send me an email at `jeanchristophe.bach@inria.fr`